REMARKS

This is a response to the non-final Office Action mailed December 26, 2007.

Applicants thank the Examiner for the telephone interview of January 30, 2008. A separate paper regarding an "Interview Summary" is being filed herewith. The discussion below is provided in part to clarify issues regarding the claims discussed during the interview, as well as to provide a complete response to the pending Office Action.

Claims 1, 4, 7, 8, 10, 13, 20, 27, 28, 32, 33, 36, 37, 40, 41, 44, 45, 47, 52, 57, 65 and 68 are amended, and claims 81 and 82 are new. No new matter is presented. Claims 2, 3, 12, 18, 25, 26, 29, 34, 35, 59, 63, 66, 71 and 72 are cancelled. Example support for the claims is as follows: claim 1 (Fig. 2, client 60; Fig. 3, step 112); claim 4 (Fig. 2, plug-in 62, p.6, lines 23-25); claim 7 (see support for claims 1 and 4); claim 8 (p.12, lines 1-8 and items (2) and (3)); claim 10 (Fig. 8, step 466); claim 20 (see support for claim 8); claim 28 (see support for claim 1); claim 36 (same as claim 10); claim 40 (see support for claim 8); claim 44 (claim 28); claim 57 (p.10, lines 10-12); claim 68 (p.5-6, bridging sentence); claim 81 (p.11, bottom; p.29, lines 2-4); and claim 82 (same as claim 4).

Regarding paragraph 2 of the Office Action, claim 65 has been amended to add a period.

Regarding paragraph 5 of the Office Action, claims 33, 35, 36, 55-58, 60-62, 64, 65, 86, 77 and 78 have been rejected under 35 U.S.C. 102(b) as being anticipated by Wu et al. (US 5,987,256) (Wu). Applicants respectfully traverse this and the other rejections.

Wu uses a precompiler which translates standard HTML, Java or other programs to display-oriented language codes for use by a thin client platform (col. 2, lines 56-61, col. 3, lines 6-12). A compile code rendering engine 12 is a relatively compact program which runs on an end user thin platform, which includes a microcontroller and limited memory (Fig. 1, col. 4, lines 11-23). The compile code rendering engine 12 translates a compiled code data set which is received from a compile code data source 13 (Fig. 1) into a stream of data suitable for a display. Wu's device is unsuited for storage and execution of an HTML rendering program, Java virtual machine or other standard rendering engine (Wu, abstract).

Applicants' claim 33 sets forth in part "receiving a request for particular content, said request is received at a server from a web client in which a plug-in is embedded" and "in response to said request, compiling said first code to create executable code for said plug-in to said web client." In contrast, Wu's compile code rendering engine 12 is not a web client in which a plug-in is embedded. Instead, Wu teaches that the compile code rendering engine 12 is the sole software which is provided on the end user thin platform. Thus, even if, arguendo, the compile code rendering engine 12 is a web client, there is no plug-in embedded in the web client, e.g., when the web client provides the request to the server for particular content. Similarly, since Wu does not use a plug-in in the thin client platform, Wu does not disclose or suggest compiling code to create executable code for a plug-in to a web client. Likewise, Wu does not transmit executable code from a server to a plug-in for execution by the plug-in.

Regarding the use of applets, mentioned by Wu at Fig. 8B, step 1239, applets are not compiled to executable code in response to a request from a web client (Applicants' specification, p.2, lines 3-14). Instead, applets are precompiled when they are created.

Thus, unlike the cited references, Applicants' invention reduces the processing requirements for commonly-available plug-in software for web clients, and allows code to be dynamically compiled in response to a web client request.

Claim 33 and its dependent claims are therefore patentable over the cited references.

For example, claim 36 sets forth in part "transforming said media content to an accepted format, said transforming is separate from said compiling." While Wu translates objects which are specified by full function HTML or Java to a format which is suitable for rendering in the thin client environment, this is achieved by compiling standard HTML or Java programs so they can run on the thin client platform (Wu, col. 2, lines 51-63). Wu provides no separate transforming of media content to an accepted format in addition to compiling of code. Claim 36 is therefore patentable over the cited references.

Regarding clams 55-57, the compiled code comprises elements which are identified by markup language tags, at least one of the elements defines a view template of a user interface element, and the elements comprise at least one element which defines a view class which supplies default properties, behavior, and child views which the view template instantiates. Wu fails to disclose or suggest these features. Regarding col. 6, lines 4-8 of Wu, this refers to fixing the

coordinates of graphics objects specified by HTML code on the screen of the target device, such as by word wrapping paragraphs, placing horizontal rules in particular places, choosing colors and executing other device specific processes. However, there is no disclosure or suggestion, for instance, of providing child views which are associated with a parent view.

Regarding claim 61, Wu does not provide an element that references a media file that contains a movie. Wu at col. 5, line 19-col. 6, line 7 states:

> "Using the parameters of the target device, and the parsing class structure set up after the parsing process, the algorithm does HTML rendering based on a class hierarchy adapted to the dimensions and palette of the target device (step 840). This fixes the coordinates of all the graphic objects specified by the HTML code on the screen of the target device. For example, the paragraphs are word wrapped, horizontal rules are placed in particular places, the colors are chosen, and other device specific processes are executed."

Thus, the graphic objects refer to paragraphs, horizontal rules, colors and the like. There is no indication by Wu that the graphic object can include a movie. The Examiner is requested to substantiate this assertion. In fact, as mentioned, the thin client of Wu is unsuited for storage and execution of an HTML rendering program, Java virtual machine or other standard rendering engine (Wu, abstract). Certainly, Wu's device having a "minimum set of resources" (Wu, abstract) cannot play a movie if it cannot even execute standard HTML or Java and it instead processes graphics objects in the form of "simplified graphics primitives" (col. 5, lines 5-7). Claim 61 is therefore patentable over the cited references.

Similarly, regarding claim 62, the use of a .SWF file as claimed cannot be considered to be a graphic object within the ordinary meaning of the claim language.

Regarding claim 68, a connection to a web service which is external to a server is defined by an element which is identified by a markup language tag (per claim 55), where the server receives a request for content, compiles code and transmits executable code (per claim 33). The cited passages at cols. 5 and 6 of Wu provide no teaching in this regard. Instead, the thin platform of Wu only communicates with a server 22 (Fig. 2), which in turn is coupled to a developer workstation 20. There is simply no disclosure or suggestion of a connection to a web service which is separate from a server as claimed. Applicants' approach thus provides syntax for automatically connecting user-interface items with back-end data sources and services, and is not limited to use of standard user

interface primitives like "window," "button," "text," "scroll bar," and so on (Applicants' specification, p.5, lines 13-19). Claim 68 is therefore patentable over the cited references.

Regarding paragraph 22 of the Office Action, claims 1, 3-10, 13, 21-24, 26-32, 37-47, 51-54, 67, 69, 70 and 79 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Wu in view of Davis et al. (US 6,643,696) (Davis).

Davis provides a process for tracking client interaction with a resource such as a web page downloaded from a server. In particular, an HTML document is downloaded from a server (e.g., server A, Fig. 3), then images specified by first embedded URLs are downloaded. A second embedded URL in the document points to a first executable program that runs on a server (e.g., server B). A third embedded URL in the document points to a second executable program that is downloaded to and runs on the client. When the images specified by the first embedded URLs are downloaded, the first executable program on the server runs. The server can capture identifying information from the client. The second executable program determines the current time when it initializes, and the current time when the user leaves the HTML document. The elapsed time is then reported to the server (col. 5, line 40-col. 6, line 16).

Regarding claim 1, the cited references do not disclose or suggest at least "receiving a request from a user device for particular content, said request is received at a server" and "accessing said data at said external data source based on said one or more source files which define said connection to said external data source, said server performs said accessing." Thus, with Applicant's invention, the same server that receives a request from a user device accesses data at an external data source. In contrast, Davis uses a server A to provide a requested web page to a client, and a server B which receives tracking information from the tracking program executing at the client (Davis, Fig. 3). The server A which provides the requested web page does not access data at an external data source (e.g., server B or other external source) based on one or more source files in a markup language description of requested content.

Furthermore, it would not be obvious to combine Wu and Davis since Davis is concerned only with tracking a user's activities but is not concerned with providing any content to the user via a user interface. Instead, Davis's system operates as a background process and is therefore not apparent to the user. Further, Davis is not concerned with providing requested content. In contrast,

Wu's sole purpose is providing requested content. Further, Davis's system is meant for use on a general purpose computer and would not be suitable for use on Wu's thin client. A person skilled in the art looking for ways to enhance Wu's technique of providing requested content would therefore not attempt to combine Davis with Wu as asserted.

Specifically, regarding the asserted motivation to combine Wu and Davis at p.8 of the Office Action, (first par.): "it would have been obvious to combine Davis with Wu for the benefit of utilizing more complex content on a thin-client device," this is counterintuitive as Wu is specifically concerned with using a minimum set of resources (Wu, abstract) in applications such as TV set top boxes, held devices and DVD players (col. 1, lines 37-43), which do not have the processing power of a general purpose computer, and therefore do not have additional resources available for handling the tracking program of Davis. Wu's goal is to reduce, not add, complexity.

In contrast, Applicants' approach allows developers to easily and economically create network-aware applications with rich, interactive user-interfaces by accessing data from external data sources such as databases, directories and web services (Applicants' specification, p.5, lines 9-11; p.5-6, bridging sentence).

Claim 1 and its dependent claims are therefore patentable over the cited references.

For example, claim 4 sets in part forth that a rendering entity at a user device executes executable code, where the rendering entity is a plug-in to a browser, and the plug-in is embedded in the browser before a request. Thus, the plug-in is already embedded when a request is made for content. The Office Action asserts that Wu's mention of an applet is a plug-in as claimed. This issue was discussed in the telephone interview. However, a common definition of an applet is "a program that is downloaded over the Internet and executed on the recipient's machine" (Microsoft Computer Dictionary (2002), see attachment). Accordingly, an applet is not present in a browser until after content is requested and delivered. Further, the applet does not execute code since the applet itself is a program that is executed by another entity such as a browser. The amended language is believed to distinguish Applicants' use of a plug-in over any use of an applet by Wu. Claim 4 is therefore patentable over the cited references.

Claim 7 similarly sets forth that a request for content is received at a server from a browser in which a rendering entity is present as a plug-in to the browser. Again, unlike Wu, the plug-in is

present when the request for content is made and is therefore not delivered with requested content. Claim 7 is therefore patentable over the cited references.

Claim 8 sets forth in part: "providing a reference in said mark-up language description to a media file which contains said media content, said media file is external to said mark-up language description" and "a rendering entity at said user device renders said media content on said user interface when said media file is referenced when said executable code is executed." In contrast, the cited references provide no disclosure or suggestion of providing a reference to a media file with media content, and rendering the media content when the media file is referenced. Instead, Davis provides no teaching regarding rendering content and is only concerned with tracking. Also, Wu provides graphics objects in the form of simplified graphics primitives and does not reference any media file as claimed. Claim 8 is therefore patentable over the cited references.

Claim 10 sets forth in part "transforming said media file to an accepted format before said transmitting of said media file, said transforming is separate from said compiling." Wu translates a first data set (e.g., HTML data) to a second data set (e.g., compiled HTML) which is adapted for execution on a target device (Fig. 6 and col. 2, lines 47-50). Thus, this transforming is the same as compiling, so there is no separate transforming which is separate from compiling. Claim 10 is therefore patentable over the cited references.

Claim 21 is patentable for the reasons discussed in connection with claim 1. Specifically, claim 21 involves the same server receiving a request for content, acquiring data from an external data source, compiling content to create executable code, and transmitting the executable code to a client. In contrast, Davis uses a server A to provide a requested web page to a client, and a server B which receives tracking information from the tracking program executing at the client (Davis, Fig. 3). The server A which provides the requested web page does not access data at an external data source (e.g., server B or other external source) based on a markup language description of requested content. Claim 21 and its dependent claims are therefore patentable over the cited references.

For example, claim 27 is patentable for the reasons discussed above in connection with claim 36.

Claim 28 is patentable for the reasons discussed above in connection with claims 1 and 7. In particular, Wu and Davis do not provide executable code and a media file to a plug in embedded in a browser, where the plug in renders content which is requested by the browser based on the

executable code and the media file. As mentioned, Wu's use of an applet does not provide a plug-in which is embedded in a browser when the browser requests content since the applet is only provided later with the content. Further, the applet itself does not render content since the applet is a program that is executed to render content. Claim 28 and its dependent claims are therefore patentable over the cited references.

For example, claim 32 is patentable for the reasons discussed above in connection with claim 36.

Claim 37 and its dependent claims are patentable over the cited references for the reasons discussed previously in connection with, e.g., claims 1 and 28.

For example, claim 40 is patentable for the reasons discussed above in connection with claim 36.

Claim 41 and its dependent claims are patentable over the cited references for the reasons discussed previously in connection with, e.g., claims 1 and 4.

Claim 45 and its dependent claims are patentable over the cited references for the reasons discussed previously in connection with, e.g., claims 1 and 4.

For example, claim 47 is patentable for the reasons discussed above in connection with claim 36.

Regarding paragraph 64 of the Office Action, claim 11 has been rejected under 35 U.S.C. 103(a) as being unpatentable over Wu in view of Russell (US2002/0069420).

This claim is patentable at least by virtual of its dependence on an independent claim which is patentable as discussed herein. Further, claim 11 sets forth that compiling (of a mark-up language description of particular content to create executable code) and transmitting (of the executable code) are only performed if authenticating is successful, where different types of authenticating are provided for different types of content and/or for each item of content. Russell is concerned with delivering content such as movie files over a network. However, there is no disclosure or suggestion of allowing compiling based on authenticating as claimed. Further, a person of ordinary skill in the art would not be led to combine these references as suggested because the thin client of Wu can only handle simplified graphics primitives, but cannot handle movie files, music files, or video games files with which Russell is concerned (par. 4).

Wu refers to running HTML and Java programs on a TV set top box, handheld device, or digital video disk player (col. 2, lines 59-63). However, Wu's rendering engine 12 (Fig. 1) is a separate component which operates in a separate surf_HTML mode (col. 17, lines 3-6). There is no indication that the surf_HTML mode provides video and audio data, for instance, to the device. It can only be concluded that the TV set top box or digital video disk player, for instance, receive video and audio data from a conventional television signal or a DVD, which do not use a markup language. Claim 11 is therefore clearly patentable.

Regarding paragraph 66 of the Office Action, claims 14-17, 19, 20, 74 and 75 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Wu in view of Wagner (US6,085,224).

Regarding claim 14, Wagner is concerned with detecting hidden data such as cookies in a data stream. Wagner states at col. 15, line 61 to col. 16, line 15 that embedded commands can be used to activate or execute a program or applet. Programs which may be activated by an embedded command include Java script programs. For example, an HTML tag can be used to invoke a Java applet. The Examiner assets that it would be obvious to modify Wu by including scripting language embedded within a markup language file. However, Wu teaches against the proposed combination since Wu sets forth that a translation process is selected according to the identified object specifying language (e.g., HTML or Java) (col. 2, lines 31-37). Further, Wu clearly provides separate compiling processes for HTML (Figs. 3 and 6) and Java (Figs. 4 and 9).

In particular, Wu states: "According to other aspects of the invention, the step of translating the first data set includes first identifying the object specifying language of the first data set from among a set of object specifying languages, such as HTML and JAVA. Then, a translation process is selected according to the identified object specifying language." (col. 2, lines 31-37). Further, Wu at col. 19, lines 35-39 states: "Thus, a standard object file, such as an HTML or JAVA image, is input online 1300 to a compiler 1301 which runs on a standard computer 1302. The output of the compiler on line 1303 is the compiled bitmap, compiled HTML or compiled JAVA formatted file."

Thus, a single "object specifying language" and associated translation process are used. Regarding the assertion in the Office Action that it would be obvious to embed script languages within a markup language file, recall that Wu's thin client device is "unsuited for storage and

execution of the HTML rendering program, JAVA virtual machine, or other rendering engine for the standard" (Wu, abstract). Thus, a person of ordinary skill in the art would not be led to try to add additional features to Wu's device. Accordingly, claim 14 and its dependent claims are patentable over the cited references.

For example, claim 20 is patentable for the reasons discussed above in connection with claim 36.

Regarding paragraph 74 of the Office Action, claims 48-50 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Wu and Davis in view of Wagner.

Claim 48 is patentable for the reasons discussed above in connection with claim 14.

Regarding paragraph 78 of the Office Action, claims 73, 76 and 80 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Wu and Davis in view of Ausems et al. (US 2003/0013483). Ausems merely indicates that a Flash player can be used on a handheld communication device. As discussed in the telephone interview, the invention of claim 78 does not involve a Flash player alone, but refers to the fact that a Flash player is a plug-in to a browser and executes executable code in the manner set forth in claims 1 and 4, e.g., where the executable code is created by compiling a markup language description of requested content. The use of a Flash player as a plug in to a browser, in a context where the browser receives executable code in response to a request for content, provides the important advantage of adding additional functionality to the browser. This is an important distinction that is not disclosed or suggested by the cited references. In contrast, Wu cannot provide such additional functionality since Wu does not use a general purpose computer which can accommodate a Flash player, and instead is specifically concerned with a thin client device. Further, Wu does not use a plug-in as discussed in connection with claim 4. Claim 73 (and similarly, claim 80) is therefore clearly patentable.

New claims

New claim 81 is patentable as the cited references do not provide an object in executable code which identifies a name and/or format of media content, and provide the name and/or format

via a user interface when the media content is rendered. Claim 82 is patentable for the reasons discussed in connection with claim 4.

Conclusion

In view of the above, each of the pending claims is believed to be in condition for immediate allowance. The Examiner is therefore respectfully requested to pass this application on to an early issue.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: February 1, 2008

By: /Ralph F. Hoppin/
Ralph F. Hoppin
Reg. No. 38,494

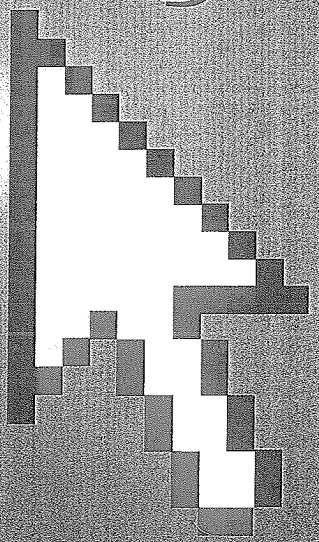
VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105-4206
Telephone: (415) 369-9660, x214
Facsimile: (415) 369-9665

Microsoft

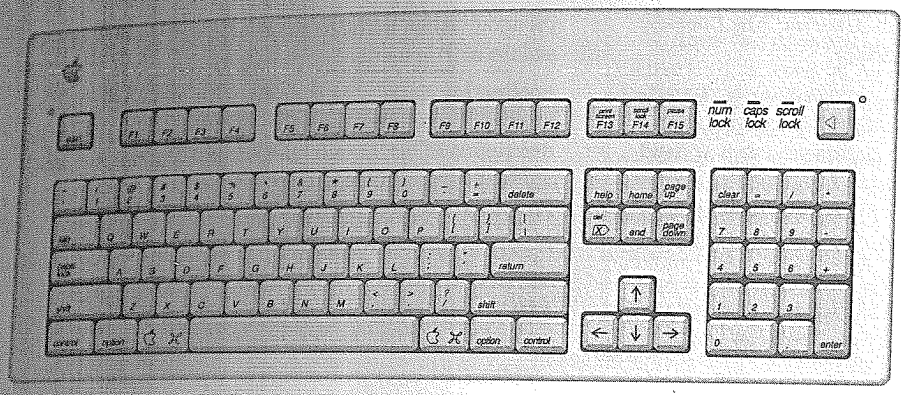
OVER
10,000
ENTRIES

Microsoft® Computer Dictionary

Fifth Edition



- *Fully updated with the latest technologies, terms, and acronyms*
- *Easy to read, expertly illustrated*
- *Definitive coverage of hardware, software, the Internet, and more!*



Apple Extended Keyboard.

Apple Filing Protocol *n. See* AFP.

Apple key *n.* A key on Apple keyboards labeled with an outline of the Apple logo. On the Apple Extended Keyboard, this key is the same as the Command key, which functions similarly to the Control key on IBM and compatible keyboards. It is generally used in conjunction with a character key as a shortcut to making menu selections or starting a macro.

Apple Macintosh *n. See* Macintosh.

Apple Newton *n. See* Newton.

AppleScript *n.* A script language developed by Apple Computer, Inc., for Macintosh computers running under the Mac OS to execute commands and automate functions. *See also* script.

AppleShare *n.* A file server software developed by Apple Computer, Inc., that works with the Mac OS and allows one Macintosh computer to share files with another on the same network. *See also* file server, Mac OS.

applet *n.* A program that can be downloaded over the Internet and executed on the recipient's machine. Applets are often written in the Java programming language and run within browser software, and they are typically used to customize or add interactive elements to a Web page.

AppleTalk *n.* An inexpensive local area network developed by Apple Computer, Inc., for Macintosh computers that can be used by Apple and non-Apple computers to communicate and share resources such as printers and file servers. Non-Apple computers must be equipped with AppleTalk hardware and suitable software. The network uses a layered set of protocols similar to the ISO/OSI reference model and transfers information in the form of packets called frames. AppleTalk supports connections to other AppleTalk networks through devices known as bridges, and it supports connections to dissimilar networks through devices called gateways. *See also* bridge, frame (definition 2), gateway.

AppleTalk Phase 2 *n.* The extended AppleTalk Internet model designed by Apple Computer, Inc., that supports multiple zones within a network and extended addressing capacity.

AppleWorks *n.* A suite of productivity applications, formerly known as ClarisWorks, distributed by Apple Computer, Inc., and shipped on the iMac computer. AppleWorks/ClarisWorks is an integrated product that includes support for word processing, spreadsheets, databases, drawing, painting, charting, and the Internet.

appliance *n.* **1.** *See* server appliance. **2.** *See* information appliance. **3.** A device with a single or limited purpose with functionality. This functionality is similar to a simple consumer appliance.

appliance server *n.* **1.** An inexpensive computing device used for specific tasks including Internet connectivity or file-and-print services. The server is usually easy to use but does not possess the capabilities or software of a typical server for general office use. **2.** *See* server appliance.

application *n.* A program designed to assist in the performance of a specific task, such as word processing, accounting, or inventory management. *Compare* utility.

application binary interface *n.* A set of instructions that specifies how an executable file interacts with the hardware